

A METHOD FOR USER VARIATION OF A MEASUREMENT PROCESS

FIELD OF THE INVENTION

This invention relates to the field of software control of a measurement process and in particular to a method for allowing a user of the measurement process to modify the process.

BACKGROUND OF THE INVENTION

Many measurement systems and processes are controlled by software running on a computer. Some tasks, called standard tasks, performed by a measurement system may be common to many users, so software to perform those tasks is often supplied with the measurement system, or by a third party vendor. Other tasks may be highly complex but, if they are to be repeated many times, considerable effort may be invested in producing the software. The software may be supplied as source code or as executable code. Often, however, a user may need to perform a task that is a variation of a standard task. For example, the user may wish to modify selective behavior of the process, modify data at selected points within the process, or add additional functionality to the process. The user may be a person or a machine.

Adding this flexibility to a commercial measurement system involves direct and indirect costs to the producer of the system. For example, proprietary knowledge of the measurement process will be revealed if source code or detailed documentation is provided the user. Production and distribution of documentation adds a direct cost, as does the provision of

1 technical support. There is an increased need for technical support if the user
2 is required to understand all parts of a complex measurement process.
3 Further, if the user is permitted to modify the measurement process without
4 sufficient safeguards, the process may be invalidated. This increases the
5 need for technical support and may also make the user dissatisfied with the
6 measurement system.

7 Prior measurement systems have attempted to address some of these
8 needs. For example, the user may be supplied with the source code for the
9 standard tasks and allowed to modify and recompile the code. This has
10 several limitations. Firstly, the user must be instructed on how to make the
11 modifications and recompile the code. Secondly, the user must purchase the
12 tools to do this. Thirdly, the user may introduce errors in the code for the
13 standard task, which increases the amount of technical support required by
14 the user. Fourthly, any proprietary methods contained within the source code
15 for the standard task will be disclosed.

16 In a variation of this approach, the standard task is separated into a
17 number of smaller tasks or steps. Each step is implement by a 'chunk' of
18 code. The user does not modify any of the standard chunks of code, but
19 writes additional code that utilizes the standard chunks of code. The user
20 may be provided with a description of the interface to each chunk of code. An
21 example of this approach is a 'block diagram' or 'visual' design tool, in which
22 each step is implemented as an independent block. The blocks may have
23 parameter and data interfaces and may be linked together. The user may be
24 provided with a template to develop his, or her, own blocks. This approach
25 can reduce the likelihood of a standard code chunk being modified in error,

1 and does not necessitate the release of source code for the standard blocks.
2 However, each chunk represents a step in a standard process, and the
3 chunks must be linked together in a framework. This requires that the user
4 understand the behavior of each of the many chunks, and so generates a
5 requirement for extensive documentation. This documentation may reveal
6 proprietary methods. In addition the approach still permits the overall process
7 to be modified in error if the framework is modified. As a consequence, the
8 measurement may be invalidated and there is still a need for significant
9 technical support from the supplier.

10 A further approach is the use of a 'plug-in' code module. A plug-in
11 code module is code that interacts with an application to provide specialized
12 data processing. For example, web browsers have plug-in code modules for
13 receiving streamed audio and visual data and presenting it to a user. Audio
14 processing and graphics programs have plug-in code modules called filters,
15 which allow data to be manipulated (transformed) in a custom manner and
16 passed back to the program. This feature allows third party vendors and
17 users to add additional features to widely used applications. However, the
18 use of plug-in code modules has been restricted to custom data manipulation;
19 they do not allow the user to modify the operation of the program.

20 A further approach is the use of inheritance in which objects or
21 components are provided for standard subtasks, and the user can use these
22 as a basis for defining more complex processes.

23 A still further approach is parameterization, in which the user is
24 provided with an interface and may adjust parameter values or select between

1 alternative subtasks. However, this approach does not provide a capability for
2 the user to add new sub tasks.

3 A still further approach is the use of templates or code generators;
4 however, these have the same limitation as block diagram tasks described
5 above.

6 In view of the above, there is an unmet need for a method that allows a
7 user to:

- 8 1. modify selective behavior of the process,
- 9 2. modify data at selected points within in the process, and
- 10 3. add additional functionality;

11 while at the same time:

- 12 1. not disclosing proprietary source code,
- 13 2. not requiring documentation of the whole process, and
- 14 3. preventing the user from invalidating the process.

15 **SUMMARY OF THE INVENTION**

16 The invention provides a method by which a user can modify or interact
17 with a software process at one or more specified points within the code.
18 These points are called variation points and are inserted by the designer at
19 points in the code where a designer anticipates that a user may want to
20 interact with or modify the process. In operation, when a variation point is
21 reached, control passes to a user-defined process or subsystem. The user-
22 defined subsystem performs one or more actions, including modification of
23 measurement data, control of a device being tested and variation of numerical
24 and control parameters defining the measurement process. Control may then
25 be passed back to the measurement process. Subsequent operations

1 performed by the measurement process may be affected through the
2 numerical and control parameters.

3 4 **BRIEF DESCRIPTION OF THE DRAWINGS**

5 The novel features believed characteristic of the invention are set forth
6 in the claims. The invention itself, however, as well as the preferred mode of
7 use, and further objects and advantages thereof, will best be understood by
8 reference to the following detailed description of an illustrative embodiment
9 when read in conjunction with the accompanying drawing(s), wherein:

10 **Figure 1** is a diagrammatic representation of a measurement system,

11 **Figure 2** is a sequence diagram for one embodiment of the invention,
12 and

13 **Figure 3** is a graphical representation of the structure of a computer
14 program in accordance with one embodiment of the invention.

15 **Figure 4** is a flow chart depicting the method by which a user of the
16 measurement process can cause a variation in the measurement process in
17 accordance with one embodiment of the invention.

18 **DESCRIPTION OF THE INVENTION**

19 While this invention is susceptible of embodiment in many different
20 forms, there is shown in the drawings and will herein be described in detail
21 one or more specific embodiments, with the understanding that the present
22 disclosure is to be considered as exemplary of the principles of the invention
23 and not intended to limit the invention to the specific embodiments shown and
24 described. In the description below, like reference numerals are used to

1 describe the same, similar or corresponding parts in the several Views of the
2 drawings.

3 Measurement and test equipment are utilized in many applications. A
4 diagrammatic representation depicting the use of a measurement system is
5 shown in **Figure 1**. Referring to **Figure 1**, the measurement system 102
6 includes measurement process software or code 104, stored in a computer
7 readable medium, and a physical interface 114. Additional interface
8 components (not shown) may also be used. The physical interface 114
9 functions to connect the measurement system 102 to a device under test 106.
10 Through the physical interface 114, the measurement system 102 may
11 provide signals 108 that control the inputs to the device under test 106, and
12 signals 111 that control the configuration of the device under test. The
13 measurement system receives output signals 110 that indicate the properties
14 or behavior of the device under test, and signals 112 that indicate the
15 configuration of the device under test. The output signals 110 may be
16 provided by detectors, such as sensors or probes.

17 The measurement process software 104 operates to control the
18 measurement system, including sequencing of input and output, control of
19 configuration and analysis of data.

20 An example of where sophisticated measurement equipment is used is
21 the testing of telecommunications devices such as cellular handsets or optical
22 multiplexers. These devices are highly complex and have many modes of
23 operation. Consequently, complex measurement equipment is required to
24 verify that the devices operate according to their specifications. The test
25 equipment must generate a variety of test signals and analyze the responses

1 to those signals. With devices of such complexity, it is often not practical for a
2 user to understand the detailed operation of the measurement equipment.
3 However, the user may still wish to modify or customize certain portions of the
4 measurement process to add addition functionality to the process.

5 The invention provides a method by which a user of the measurement
6 system can modify or interact with the measurement process at one or more
7 specified points within the measurement process software. These points are
8 called variation points and are inserted by the designer at points in the code
9 where the designer anticipates that a user may want to interact with or modify
10 the process.

11 In operation, when a variation point is reached, control passes to a
12 user-defined process or subsystem. The user-defined subsystem performs
13 one or more actions, including modification of measurement data, control of
14 the device being tested and variation of numerical and control parameters
15 defining the measurement process. Control may then be passed back to the
16 measurement system. Subsequent operations performed by the
17 measurement system may be affected through the numerical and control
18 parameters.

19 **Figure 2** is a sequence diagram for one embodiment of the method of
20 the invention. In the diagram, the downward vertical direction indicates
21 advancing time, while each vertical column represents an instance of a
22 process or a module. In the simple exemplary embodiment depicted in
23 **Figure 2**, there are three processes, namely, calling function 202, standard
24 measurement process 204 and process modification module 206. The
25 sequence is started by the calling function 202. The time line of the calling

function is denoted by the vertical broken line 208, the time line of the standard measurement process is denoted by 210 and the time line of the process modification module is denoted by 212. It is to be understood that the time lines shown relate to the modification of the measurement process, and that many other operations are typically performed in a measurement process. The arrow 214 denotes a call from the calling function to the standard measurement process to register the interface of the process modification module with a function named `make_variation` in the measurement process. This may take the form, for example, of the function call

```
register_variation(interface:Process_Modification)
```

In this embodiment, this indicates that when the variation point is reached, the interface should be serviced by a function within the `Process_Modification` module.

The arrow 216, denotes invocation of the standard measurement process, and may take the form of the function call

```
run().
```

The duration of the standard measurement process is denoted by the block 218.

At some point in the standard measurement process, a variation point is reached and, as indicated by the arrow 220, control is transferred to the `Process_Modification` module. Parameters are passed using the interface registered at 214. The call may take the form

```
make_variation(variationID:String, data:String).
```

1 In this example, the arguments of the `make_variation` function
2 comprise an identification string and a data string. However, other types of
3 parameters may be used. The data string may contain measurement data,
4 numerical parameters and control parameters. The duration of the
5 `Process_Modification` module is indicated by the block 222. After the
6 appropriate modifying action has been taken, control returns to the
7 measurement process as indicated by the arrow 224. Any element of the
8 data string may be changed, and passed back to the measurement process.
9 The portion of the measurement process subsequent to 224 may thus be
10 modified. For example, alternative process 226 may be implemented rather a
11 portion of the block 218, dependent upon a parameter set by the variation
12 function.

13 Finally, when the measurement process 218 is completed, control is
14 returned to the calling function, as indicated by the arrow 228.

15 In general, multiple variation points are inserted into the code for the
16 measurement process. In one embodiment, each variation point is associated
17 with an identifier and each identifier is associated with a process modification
18 function, ensuring that the appropriate modification action will be taken. In a
19 further embodiment, each variation point is associated with a function name,
20 e.g. `make_variation1`, `make_variation2`, ...etc. The user may provide
21 functions with the appropriate names, which are dynamically linked with the
22 measurement code when it is loaded.

23 Each variation point can use a separate interface or each variation
24 point can use the same interface with a marker or parameter or key.

1 In this example, interaction between the measurement system 102 in
2 **Figure 1** and the device under test has not been shown. The process
3 modification module may interact with the device under test. In one
4 embodiment, this interaction is made independent of the measurement
5 system. In a further embodiment the physical interface of the measurement
6 system is used to facilitate the interaction. In a still further embodiment,
7 components of the measurement process are utilized to interact with the
8 device under test.

9 In the example described above, the standard (unmodified)
10 measurement process is realized by a single software system. **Figure 3(A)** is
11 a simplified UML (Unified Modeling Language) class diagram showing the
12 software classes associated with measurement system 302. The class
13 diagram is a graphical representation of the structure of the code associated
14 with the modification process. Internal to the measurement system are a
15 number of functions 304. These include the function

16 `+register_variation(in`
17 `interface:*Process_Modification)`

18 which indicates that the call to the function `make_variation` in the
19 measurement process should be serviced by the corresponding function in
20 the interface of the modifying module `Process_Modification`, and the
21 function

22 `+run()`

23 which invokes the measurement process, and has no calling arguments. In
24 the UML, the symbol '+' is used to denote that the function is accessible
25 outside of the class, i.e. that it is a public function.

1 The arrow 306 denotes a realization relationship between the
2 measurement system 302 and an interface named IVariation, 308. This
3 indicates that the interface is realized by the measurement system. The
4 interface includes a function 310 that returns a string value, as indicated by
5 the notation " : String". The function has two inputs, an identifier string
6 variationID and a data string data. The data string is also an output
7 since it can be modified by the process modification component.
8 Alternatively, the variation can be made through a value returned by the
9 function.

10 **Figure 3(B)** is a simplified UML class diagram showing the software
11 classes associated with process modification subsystem 312. The subsystem
12 includes a function `make_variation` 314 that returns a string.

13 The `register_variation` function call determines that this is the
14 function to be used to service the function call at the variation point. The
15 process modification subsystem 312 supports the interface 316, which
16 indicates that the process modification subsystem is called using the interface
17 IVariation.

18 If more than one process modification functions are to be used, the
19 calling arguments in 310 and 314 include a variation point identifier,
20 `variationID`, so that the process modification subsystem can select action
21 appropriate to the variation point at which the call was made.

22 The interaction between the measurement process and the process
23 modification module is determined by the interface. In one embodiment, the
24 interface is specified at a binary (bit) level. The process modification module

1 may be written in any computer language and run on any computer provided
2 that it supports the interface.

3 A variety of methods for performing the call-out to the process
4 modification module will be apparent to those skilled in the art. For example,
5 in one embodiment, the process modification module is provided as a
6 dynamically linked library (DLL) of functions that runs on the same computer
7 and in the same memory space as the measurement process. The module
8 could alternatively be provided as a run-time DLL that runs on the same
9 computer but in a separate memory space. As further options, Connection
10 Point Interfaces (Visual Basic Events) or an Event Server could be used as an
11 interface mechanism.

12 A Component Framework may be utilized, which supports the creation
13 of runtime instances of Components, and provides services for Components
14 to be discovered, communicate, persist, etc. For example, the process
15 modification component may be provided as a Component Object Module
16 (COM) DLL file or as a COM executable file that runs on the same computer
17 but in a memory space separate from the measurement process. The
18 measurement process may instantiate a predetermined COM interface
19 definition and binary data specification. Alternatively, the measurement
20 process may invoke one or more COM interfaces passed into the
21 measurement process.

22 Similarly, the Common Object Request Broker Architecture (CORBA)
23 provides a set of specifications that define a central inter-component
24 communication mechanism and a set of service components to provide
25 common services. The call-out may use one or more CORBA-based servers.

1 This provides the additional functionality of allowing the process modification
2 component to run on a separate computer, which may be remote from the
3 measurement system and connected to it via a network.

4 The process modification module may be implemented as an
5 Enterprise Java Bean (EJB).

6 The invocation of the process modification module and the associated
7 parameter passing may be implemented using Simple Object Access Protocol
8 (SOAP), which is an XML-based format for specifying method invocations
9 between computer systems (XML or eXtensible Markup Language is a
10 language for facilitating communication between Components). SOAP is
11 completely independent of any platform, operating system or programming
12 language and can easily be used over the Internet with the widely used HTTP
13 protocol. Microsoft web service could similarly be used as the interface
14 mechanism.

15 Other methods for performing the call-out to the process modification
16 module will be apparent to those of ordinary skill in the art.

17 Use of the above techniques allows the user to develop the process
18 modification module using a computer architecture and computer language of
19 their choice.

20 The process modification module may interact with the measurement
21 process in various ways. It can perform additional operations or transforms
22 on the data. This can include modification of data sent to the device under
23 test, modification of the data received from the device under test or
24 modification of data derived from that received. In addition, it can modify the
25 measurement procedure by modifying numerical parameters and by modifying

1 control parameters. The control parameters are used to determine
2 subsequent flow through the measurement process or to determine options or
3 alternatives.

4 The process modification component can pass instruction codes to
5 send to the device under test or to other instruments.

6 The process modification component can interact with the device under
7 test to alter its configuration.

8 A flow chart depicting the method by which a user of the measurement
9 process can cause a variation in the measurement process is shown in
10 **Figure 4.** Referring to **Figure 4**, the method starts at start block 402. At
11 block 404 the user determines what variations to the measurement process
12 are desired. These variations preferably takes place at variation points within
13 the measurement process, as determined by the designer of the
14 measurement process, thereby preventing the user from invalidating the
15 measurement. At block 406, the user generates at least one code module
16 containing functions defined by the user. The code module may be written by
17 the user, constructed from existing components or produced by editing a code
18 template. The user-defined functions are designed to cause the desired
19 variations. At block 408 the user-defined functions are associated with the
20 variation functions called at the variation points within the measurement
21 process, in order that control is passed to the appropriate user-defined
22 function when the variation point is reached. For example, this step may
23 require simply placing the code module in a predetermined directory on the
24 computer or registering the functions (or an interface including the functions)
25 with an operating system. At block 410 the measurement process is

1 executed. When the measurement process reaches a variation point, the
2 user-defined function associated with that variation point is called, thereby
3 allowing the measurement process to be varied. The method ends at
4 termination block 412.

5 While the invention has been particularly shown and described with
6 reference to a preferred embodiment, it will be understood by those skilled in
7 the art that various changes in form and detail may be made therein without
8 departing from the spirit and scope of the invention.

9 What is claimed is:

0055706-01B01